计算几何

二维

```
#include<iostream>
#include<math.h>
#include<algorithm>
using namespace std;
#define ld double
typedef long long ll;
const ld Pi=acos(-1.0);
const ld eps=1e-8;
const int N=4e5+9;
int sgn(ld a){
        return a>eps?1:(a<-eps?-1:0);
}
ld Abs(ld a){
        return a*sgn(a);
}
#define V Point
struct Point{
        ld x,y;
        Point(ld _x=0,ld _y=0){
                x=_x;y=_y;
        }
        void input(){
                scanf("%lf %lf",&x,&y);
        }
        void output(){
                printf("%.5f %.5f\n",x,y);
        }
        void init(){
                if(sgn(x)==0)x=0;
                if(sgn(y)==0)y=0;
        }
        bool operator ==(const V b)const{
                return !sgn(x-b.x)&&!sgn(y-b.y);
        }
        V operator +(const V b)const{
                return V(x+b.x,y+b.y);
        }
        V operator -(const V b)const{
                return V(x-b.x,y-b.y);
        }
        V operator *(const ld b)const{
                return V(x*b,y*b);
        }
        V operator /(const ld b)const{
                return V(x/b,y/b);
        }
        ld operator *(const V b)const{
                return x*b.x+y*b.y;
        }
        ld operator ^(const V b)const{
                return x*b.y-y*b.x;
        }
}O(0,0);
ld len2(V p1){
        return p1*p1;
}
ld len1(V p1){
        return sqrt(len2(p1));
}
ld Angle(V p1,V p2){//还没检验过
        return acos((p1*p2)/(len1(p1)*len1(p2)));
}
//绕（0，0）逆时针旋转a 向量theta角度（弧度制）
V rot_PO(V a,ld theta){//还没检验过
        ld cs=cos(theta),sn=sin(theta);
        ld x=a.x*cs-a.y*sn,y=a.x*sn+a.y*cs;
        return V(x,y);
}
V rot_PO90(V a){
        return V(-a.y,a.x);
}
//绕b逆时针旋转a 向量thrta角度（弧度制）
V rot_PP(V a,V b,ld theta){//还没检验过
        V c=a-b;
        c=rot_PO(c,theta);
        return c+b;
}
ld dis_PP(V p1,V p2){
        return len1(p1-p2);
}



//Point   &   Line
bool jud_in_PLine(V p,V a,V b){
        return !sgn((p-a)^(b-a));
}
ld dis_PLine(V p,V a,V b){
        V e=a-b,c=p-a;
        return fabs((e^c)/(len1(e)));
        //return Abs(p^c/(len1(c)));
```

```
}
V find_fp_PLine(V p,V a,V b){
        V e=(b-a);e=e/len1(e);
        ld t=(p-a)*e;
        return  a+e*t;
}
V find_sm_PLine(V p,V a,V b){
        V ft=find_fp_PLine(p,a,b);
        return ft*2-p;
}


//Point   &  Line Segment
bool jud_in_PSeg(V p,V a,V b){
        Point x=p-a,y=p-b;
        return !sgn(x^y)&&sgn(x*y)<=0;
}
ld dis_PSeg(V p,V a,V b){
        V x=p-a,y=p-b,z=b-a;
        if(x*z<0)return len1(x);
        if(y*z>0)return len1(y);
        return dis_PLine(p,a,b);
}


//Line & Line | Line Segment
bool jud_cro_LineLine(V a,V b,V c,V d){
        V e1=b-a,e2=d-c;
        return sgn(e1^e2)!=0;
}
bool jud_oth_LineLine(V a,V b,V c,V d){
        V e1=b-a,e2=d-c;
        e2=rot_PO90(e2);
        return sgn(e1^e2)==0;
}
V find_cro_LineLine(V a,V b,V c,V d){//要先判断直线是否相交，以及重合
        V x=b-a,y=d-c,z=a-c;
        return a+x*((y^z)/(x^y));//no problem
}
bool jud_cro_LineSeg(V a,V b,V c,V d){
        if(jud_cro_LineLine(a,b,c,d)){
                V cp=find_cro_LineLine(a,b,c,d);
                if(jud_in_PSeg(cp,c,d)){
                        return 1;
                }
        }else if(jud_in_PLine(c,a,b)||jud_in_PLine(d,a,b)){
                return 1;
        }
        return 0;
}
bool jud_cro_SegSeg(V a,V b,V c,V d){
        if(jud_cro_LineLine(a,b,c,d)){
                V cp=find_cro_LineLine(a,b,c,d);
                if(jud_in_PSeg(cp,c,d)&&jud_in_PSeg(cp,a,b)){
                        return 1;
                }
        }else if(jud_in_PSeg(c,a,b)||jud_in_PSeg(d,a,b)){//端点在线
                return 1;
        }else if(jud_in_PSeg(a,c,d)||jud_in_PSeg(b,c,d)){//端点在线
                return 1;
        }
        return 0;

        /*

        bool SegmentProperIntersection(Point a1, Point a2, Point b1
    double c1 = Cross(a2-a1, b1-a1), c2 = Cross(a2-a1, b2-a1);
    double c3 = Cross(b2-b1, a1-b1), c4 = Cross(b2-b1, a2-b1);
    //if判断控制是否允许线段在端点处相交，根据需要添加
    if(!sgn(c1) || !sgn(c2) || !sgn(c3) || !sgn(c4)){
        bool f1 = OnSegment(b1, a1, a2);
        bool f2 = OnSegment(b2, a1, a2);
        bool f3 = OnSegment(a1, b1, b2);
        bool f4 = OnSegment(a2, b1, b2);
        bool f = (f1|f2|f3|f4);
        return f;
    }
    return (sgn(c1)*sgn(c2) < 0 && sgn(c3)*sgn(c4) < 0);
        }
        */
}



//Polygon
ld area(V a,V b,V c){
        return (b-a)^(c-a);
}
ld area(V p[],int num){
        ld ans=0;
        for(int i=2;i<num;i++){
                ans+=area(p[1],p[i],p[i+1]);
        }
        ans/=2;
        return ans;
}
//严格c在ab左边 ,  1  ,   0    ,   -1
int jud_inleft(V a,V b,V c){
        return sgn((b-a)^(c-b));
}
```

```
//判断点a在任意多边形p内（在边上==1）
int jud_in_PPoly(V a,V p[],int num){
    int wn=0;
    p[num+1]=p[1];
    for(int i=1;i<=num;i++){
        if(jud_in_PSeg(a,p[i],p[i+1])){
            return 1;//in_line;
        }
        int cmp=sgn((p[i+1]-p[i])^(a-p[i]));
        int d1=sgn(p[i].y-a.y);
        int d2=sgn(p[i+1].y-a.y);
        if(cmp>0&&d1<=0&&d2>0)wn++;
        if(cmp<0&&d2<=0&&d1>0)wn--;
    }
    if(wn){
        return 2;
    }
    return 0;
}
//判断点a在凸包p内（在边上）+二分法
bool jud_in_PConploy(V a,V p[],int num){//没验证
    p[num+1]=p[1];
    if(jud_in_PSeg(a,p[1],p[num])||jud_in_PSeg(a,p[1],p[2]))ret
    if(jud_inleft(p[1],p[2],a)<1||jud_inleft(p[num],p[1],a)<1)r
    int l=2,r=num-1,pos=2;
    while(l<r){
        int mid=l+r>>1;
        if(jud_inleft(p[1],p[mid],a)==1){
            l=mid+1;pos=mid;
        }else r=mid-1;
    }
    if(jud_inleft(p[pos],p[pos+1],a)==1)return 2;
    if(jud_in_PSeg(a,p[pos],p[pos+1]))return 1;
```

```
        return 0;
}
//以上为最基础的版子


bool cmp1(V a,V b){//这里用sgn，看情况使用，按x排序
        if(sgn(a.x-b.x)==0){
                return a.y<b.y;
        }else return a.x<b.x;
}
bool cmp2(V a,V b){//按y排序
        if(sgn(a.y-b.y)==0){
                return a.x<b.x;
        }
        return a.y<b.y;
}
bool cmp3(V a,V b){//按极角排序
        return sgn(a^b)>0;
}


/*
1:建立凸包   Andrew算法（水平序Graham扫描法），比Graham-Sacn性能更优，
*/
void get_ConvexHull(V p[],int num,V ans[],int &ansnum){
        sort(p+1,p+1+num,cmp1);
        ansnum=0;
        for(int i=1;i<=num;i++){
                while(ansnum>=2&&jud_inleft(ans[ansnum-1],ans[ansnu
                ans[++ansnum]=p[i];
        }
        int st=ansnum;
        for(int i=num-1;i;i--){
                while(ansnum>st&&jud_inleft(ans[ansnum-1],ans[ansnu
                ans[++ansnum]=p[i];
        }
        ansnum--;
}
/*
2:平面最近点对
第二种方法 O（n^longn）
用这个，验证过了
*/

ll len3(V a){
        return a.x*a.x+a.y*a.y;
}
V a[N],tmp[N];
int b[N];
void merge(int l,int r){
        int mid=(l+r)>>1,i=l,j=mid+1;//归并
        for(int k=l;k<=r;k++)
        {
                if(i<=mid&&(j>r||a[i].y<a[j].y))tmp[k]=a[i++];
                else tmp[k]=a[j++];
        }
        for(int i=l;i<=r;i++)a[i]=tmp[i];
}

//结果是长度的平方
ll dis_minPP(int l,int r){
        if(l>=r)return 1e18;
        if(l+1==r){
                if(a[l].y>a[r].y)swap(a[l],a[r]);return len3(a[l]-a
        }
        int mid=(l+r)>>1,t=a[mid].x,cnt=0;//重新排序后中位数就乱了，
        ll d=min(dis_minPP(l,mid),dis_minPP(mid+1,r));
        merge(l,r);
        for(int i=l;i<=r;i++)
                if((a[i].x-t)*(a[i].x-t)<d)//两边平方的技巧
                        b[++cnt]=i;//区间内的拉出来处理
        for(int i=1;i<=cnt;i++)
                for(int j=i+1;j<=cnt&&(a[b[j]].y-a[b[i]].y)*(a[b[j]
                        d=min(d,len3(a[b[j]]-a[b[i]]));
        return d;
}

/*
3:旋转卡壳，分别计算最远点、最左点、最右点，计算最小矩形覆盖
*/
V minRect[5];
ld get_minRect(V p[],int num){
        p[num+1]=p[1];
        if(num<=2){
                return 0;
        }
        int j=3,l=2,r=2;
        while(area(p[1],p[2],p[j])<area(p[1],p[2],p[j+1])) j=j%num+
        l=j;
        ld ans=1e18;
        for(int i=1;i<=num;i++){
                while(area(p[i],p[i+1],p[j])<area(p[i],p[i+1],p[j+1
                V A=p[i+1]-p[i];
                while(A*(p[l+1]-p[l])<0)l=l%num+1;
                while(A*(p[r+1]-p[r])>0)r=r%num+1;
                ld d=A*(p[r]-p[l])/len2(A);
                ld tmpans=area(p[i],p[i+1],p[j])*d;
                if(tmpans<ans){
                        ans=tmpans;
                        V eab=p[i+1]-p[i];
                        V ecd=rot_PO90(eab);
                        minRect[1]=find_cro_LineLine(p[i],p[i+1],p[
                        minRect[2]=find_cro_LineLine(p[i],p[i+1],p[
                        minRect[3]=find_cro_LineLine(p[j],p[j]+eab,
                        minRect[4]=find_cro_LineLine(p[j],p[j]+eab,
                }
        }
        return ans;
}
/*
4:半平面交
没验证,一半
*/
struct Line{
        Point a,b,e;
        ld ang;
        Line(Point _a=O,Point _b=O){
                a=_a;b=_b;e=b-a;
                ang=atan2(b.y-a.y,b.x-a.x);
        }
        void init(){
                e=b-a;
                ang=atan2(b.y-a.y,b.x-a.x);
        }
};
Point find_cro_LineLine(Line a,Line b){
        return find_cro_LineLine(a.a,a.b,b.a,b.b);
}
int jud_inleft(V a,Line b){
        return jud_inleft(b.a,b.b,a);
}
bool cmp4(Line a,Line b){
```

```
                if(sgn(a.ang-b.ang)==0){
                        if(jud_inleft(a.a,b.a,b.b)){
                                return 1;
                        }else return 0;
                }else return a.ang<b.ang;
}
Line stkl[N];
void HPI(Line l[],int num,V p[],int &ansnum){
                for(int i=1;i<=num;i++)l[i].init();
                sort(l+1,l+1+num,cmp4);
                int hd=1,tl=0,cnt=0;
                for(int i=1;i<num;i++){
                        if(sgn(l[i].ang-l[i+1].ang)==0)continue;
                        l[++cnt]=l[i];
                }
                l[++cnt]=l[num];
                for(int i=1;i<=num;i++){
                        while(hd<tl&&jud_inleft(find_cro_LineLine(stkl[tl],
                        while(hd<tl&&jud_inleft(find_cro_LineLine(stkl[hd],
                        stkl[++tl]=l[i];
                }
                while(hd<tl&&jud_inleft(find_cro_LineLine(stkl[tl],stkl[tl-
                while(hd<tl&&jud_inleft(find_cro_LineLine(stkl[hd],stkl[hd+
                ansnum=0;
                stkl[tl+1]=stkl[hd];
                for(int i=hd;i<=tl;i++){
                        p[++ansnum]=find_cro_LineLine(stkl[i],stkl[i+1]);
                }
}


/*
                圆的各种情况
```

```
*/




struct Circle{
        V c;
        ld r;
        Circle(V _c=O,ld _r=0){
                c=_c;r=_r;
        }
        void input(){
                scanf("%lf %lf %lf",&c.x,&c.y,&r);
        }
};
//三角形的外接圆,仅仅考虑能构造出来的情况
//三角形伸脚变两倍，b又c也*2，px=p1.x+c*len2b-b*len2c 的y
Circle get_outCircle(V p1,V p2,V p3){
    V b=p2-p1,c=p3-p1;
        ld len2b=len2(b),len2c=len2(c);
        ld d=b^c*2;
    ld ax=(c.y*len2b-b.y*len2c)/d+p1.x;
    ld ay=(b.x*len2c-c.x*len2b)/d+p1.y;
    V p(ax,ay);
    return Circle(p,len1(p1-p));
}

//三角形内接圆,仅仅考虑能构造出来的情况
Circle get_inCircle(V p1,V p2,V p3){
    ld a=len1(p2-p3),b=len1(p3-p1),c=len1(p1-p2);
    V p=(p1*a+p2*b+p3*c)/(a+b+c);
    return Circle(p,dis_PLine(p,p1,p2));
}
//直线与圆的交点
```

```
void get_cro_LineCir(Line l,Circle c,V p[],int &num){
        V fp=find_fp_PLine(c.c,l.a,l.b);
        ld dis=len1(c.c-fp);
        int t=sgn(dis-c.r);
        if(t==1){
                num=0;
        }else if(t==0){
                num=2;
                p[1]=fp;p[2]=fp;
        }else{
                num=2;
                V e=l.a-l.b;e=e/len1(e);
                dis=sqrt(c.r*c.r-dis*dis);
                p[1]=fp+e*dis;p[2]=fp-e*dis;
        }
}
//圆与圆的交点
void get_cro_CirCir(Circle c1,Circle c2,V p[],int &num){
        //两种都行
        V a=c2.c-c1.c;
        ld dis=len1(a);
        ld ang1=atan2(a.y,a.x);
        ld ang2=acos((dis*dis+c1.r*c1.r-c2.r*c2.r)/(2*dis*c1.r));
        num=2;
        a.x=c1.r;a.y=0;
        p[1]=c1.c+rot_PO(a,ang1+ang2);
        p[2]=c1.c+rot_PO(a,ang1-ang2);
//      V a=c2.c-c1.c;
//      ld dis=len1(a);
//      if(dis*dis+c1.r*c1.r<c2.r*c2.r)a=a*-1;
//      int t=sgn(dis-c1.r-c2.r);
//      if(t==1){
//              num=0;
//      }else{
//              ld d=c1.r+c2.r+dis;d/=2;
//              ld siz=sqrt(d*(d-dis)*(d-c1.r)*(d-c2.r));
//              ld h=siz*2/dis;d=sqrt(c1.r*c1.r-h*h);
//              a=a/len1(a);
//              if(t==0){
//                      num=2;
//                      p[1]=c1.c+a*d;
//                      p[2]=c1.c+a*d;
//              }else{
//                      V b=rot_PO90(a);
//                      num=2;
//                      p[1]=c1.c+a*d+b*h;
//                      p[2]=c1.c+a*d-b*h;
//              }
//      }
}

//过点与圆的切线 的交点
void get_cro_PCir(Point a,Circle c1,V p[],int &num){//这个牛逼，抄的
        ld c=len2(a-c1.c);
        ld b=max(0.0,c-c1.r*c1.r);b=sqrt(b);
        V nv=(a-c1.c)*c1.r*c1.r/c;
        V pv=rot_PO90(a-c1.c)*c1.r*b/c;
        int t=sgn(c-c1.r*c1.r);
        num=t+1;
        p[1]=c1.c+nv+pv;p[2]=c1.c+nv-pv;
}

//圆与圆的公共切线数量
int get_comTagCnt_CirCir(Circle c1,Circle c2){
        ld dis=len1(c1.c-c2.c),mi=min(c1.r,c2.r),ma=max(c1.r,c2.r);
        if(sgn(dis-mi-ma)==1)return 4;
        if(sgn(dis-mi-ma)==0)return 3;
        if(sgn(dis+mi-ma)==1)return 2;
        if(sgn(dis+mi-ma)==0)return 1;
        return 0;
}
//圆与圆的公共切线
void get_comTag_CirCir(Circle c1,Circle c2,V p[],int &num){
        int cnt=get_comTagCnt_CirCir(c1,c2);num=0;
        if(cnt==0)return;
        if(cnt==1){
                V d=c2.c-c1.c;
                d=d/len1(d)*c1.r;
                if(c1.r<c2.r)d=d*-1;
                p[++num]=d+c1.c;
                return;
        }
        if(cnt==3){
                V d=c2.c-c1.c;
                d=d/len1(d)*c1.r;
                p[++num]=d+c1.c;
        }
        ld d=len1(b.c-a.c),c,s;
        V v=(b.c-a.c)/d,v1=v*a.r,v2=v*b.r
        //还没写完
}



int main()
{


}
```